Force-Based Clustering for Transitive Identity Mapping

H. Van Dyke Parunak, Sven A. Brueckner
Soar Technology
3600 Green Court, Suite 600
Ann Arbor, MI 48105
{van.parunak, sven.brueckner}@soartech.com

ABSTRACT

In most information retrieval systems, software processes (whether agent-based or not) reason about passive items of data. An alternative approach instantiates each piece of information as an agent that actively seeks to organize itself with respect to other agents (including queries). In this approach, information objects actively self-organize. Imitating the movement of bodies under physical forces, we describe a distributed algorithm ("force-based clustering," or FBC) for dynamically clustering and querying large, heterogeneous, dynamic collections of entities. The algorithm does not simply label entities with cluster names, but moves them in a virtual space in a way that estimates the transitive closure of the pairwise comparisons. We demonstrate the operation of this algorithm on a large, heterogeneous collection of records, each representing a person. We have some information about a person of interest, but no record in the collection directly matches this information. Application of FBC identifies a small subset of records that are good candidates for describing the person of interest, for further manual investigation and verification.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence]:** Distributed Artificial Intelligence – *Multiagent Systems*
H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – *Information filtering, Retrieval models, Search process*

General Terms

Algorithms.

Keywords

Keywords are your own designated keywords. We suggest repeating the values on the three classification axes: Description, Inspiration, and Focus.

1. INTRODUCTION

Many real-world search problems require making use of inexact matches against heterogeneous data sources. Consider, for example, a person of interest (POI) who is traveling away from home. His cell phone died shortly before his departure, and he has borrowed a phone from a friend. During his travels, the POI leaves the number of the borrowed phone with someone whom he wants to contact him, but the number is partially illegible. The kind of information we might have includes a directory of cell phone numbers with the name and address of the registered user, regular telephone white pages with names, addresses, and landline phone numbers, airline reservation information with names, origin, destination, flight number, and time, and hotel registrations including name and credit card number. The names in these different sources are not represented in a consistent format, and there may be spelling errors. Identifying the POI requires computing the transitive closure of numerous relations. Here is one, but not by any means the only, route to a solution: the cell phone number leads to addresses for people who have some connection with the POI. These addresses may indicate the home area of the POI. Searching flight records for people who traveled from that area to the area where the phone number was left would generate a set of candidates for the POI, which might be further narrowed by matching against hotel registrations in hotels that are particularly near the location where the number was left.

Appears in: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013)*, Ito, Jonker, Gini, and Shehory (eds.), May, 6–10, 2013, Saint Paul, Minnesota, USA.

Constructing and reasoning over such scenarios is combinatorially prohibitive, and too slow for emergencies (such as tracking the outbreak of an epidemic or disrupting a terrorist attack) where it is critical to find the POI quickly. Our subsymbolic approach does not require such complex preparation. We instantiate each entity as a software agent in an abstract low-dimensional space (a three-dimensional torus wrapped in four dimensions). The agents compute virtual "forces" among themselves, and move in response to those forces. The transitivity of these forces brings together agents whose similarity may not be documented directly, but that are linked by a chain of similar agents.

Section 2 describes a specific information problem. Section 3 defines our algorithm. Section 4 relates it to other techniques. Section 5 reports its performance on real data. Section 6 discusses directions for extension. Section 7 concludes.

2. PROBLEM STATEMENT

Imagine the following scenario:

> An unidentified male visited a medical clinic. The individual signed in with an illegible signature, and partially illegible phone number. Before receiving attention from medical staff, the individual exited the facility. While examining other patients of the clinic, it was discovered that one of the patients showed symptoms of an influenza-like illness consistent with a potentially deadly and highly contagious virus. Staff initiated quarantine procedures to limit close contact until laboratory confirmed diagnosis of influenza was completed. In reviewing the sign-in log, staff discovered an entry which was unaccounted for. When asked, patients of the clinic could not recall any supporting information about the unidentified individual.

For the good both of the mystery patient and of the general public, it is essential to identify this person as quickly as possible.

Table 1: Databases.—A '?' indicates that the field is present in only some records of the DB.

DB name	# records	Available Fields										
		Last name	Middle name	Middle initial	First name	Street #	Street	City	State	Zip	Phone	Doc ID
CCCR	49183	X	X		X	X	X	X	X	X		X
CCTR	50001					X	X	X	X	X		X
HPA	49076	X			X		X	X	X	X		
HR	50001					X	X	X	X	X	X	
ID	49211	X	X		X	X	X	X	X	X		X
SRLU	130	X		X	X	X	X	X	X	X	X	
TR	49207	X		?	X						X	X
WP	49176	X			X	X	X	X	X	X	X	

An anonymous sponsor provided us with eight databases (DBs), containing varying combinations of name, address, phone number, and DB-specific record identifiers (Doc-IDs) for fictitious individuals, but concealing the identity of the POI. Table 1 summarizes the six databases. The total number of records is on the order of 350,000.

Cursory examination of the data indicates that some DB-specific keys are shared both within and across databases, and some names appear to be variant spellings (e.g., "Tom F. Tuk" and "Tolman Fredegar Took" share other information). Only the last two digits of the phone number are illegible, but 104 records have phone numbers that could match the available number, some associated with different names or addresses, suggesting errors in the data. The phone numbers are all in a different state than the clinic.

Our task is to develop a prioritized list of people with contact information whom authorities should contact in an effort to reach the mystery patient as quickly as possible.

3. TECHNICAL APPROACH

We first summarize the motivation and intuition for Force-Based Clustering (FBC), then describe the implementation.

3.1 Motivation

Our approach is motivated by the metaphor of physical forces. Interacting physical entities show several characteristics:

- If they are very close together, they repel one another (the physical principle of mutual exclusion).
- If they are very far apart, their interaction rapidly decreases (depending on the physical force involved, as the square or even higher powers of the separation).
- Multiple forces can contribute to interactions (e.g., gravity, kinetic linkages, electrostatic force).

We treat each record in our database as an agent. We distribute them in an abstract space, define virtual forces among them, and let them move. The intuition is that similar records will move closer to one another, pulling their neighbors with them (and thus providing transitive closure). We query the system by inserting a query record that contains what we know about the POI, letting the agents move until the system has converged, and retrieving records that end up close to the query. The closer a record is to the query, the higher we rank it in our list of persons to contact.

We emulate the features of physical movement.

- Extremely close agents repel one another, keeping similar records from collapsing to the same location, in spite of attractive forces among them.
- The decrease of interaction strength with distance means that most interactions are among nearby agents. This locality of interaction facilitates convergence of any digital algorithm for computing agent movement. Physical forces act all at once, but an algorithm must manipulate a subset of agents at each time step. Local interactions reduce the set of agents with which a given agent effectively interacts, allowing their influence to be felt in fewer steps.
- The concept of multiple forces lets us handle heterogeneous records with varying field contents. In the physical world, one force often dominates: we do not worry about electrostatic effects among planets, nor about gravitational contributions to the relative movement of charged particles. In our application, the "last-name force" can be comparable in strength to the "address force" or the "phone-number-force." Integration of these forces through agent movement allows transitive interactions among records that do not directly overlap. For example, imagine that record A has only address and DB key information (database "CCTR"), B has address and name ("CCCR"), and C has name and phone number ("TR"). The "address force" will bring A and B together, the "name" force will bring B and C together, and as a result, A and C come close together, suggesting a link between the phone number in C with the DB key in A.

3.2 Implementation

We discuss four features of our implementation: similarity computation, force definition, distributed execution, and convergence detection.

3.2.1 Similarity Computation

From the union of the fields in the databases, we define nine complex attributes in [0, 1], each derived by its own rules from one or more raw attribute fields. These rules take care of missing simple attributes (e.g., a full-name complex attribute with no middle name) and may also employ external data sources in the similarity calculation. The overall similarity between two records is a weighted sum of the component similarities.

The similarity computations and weights we assign to each complex attribute reflect our understanding about the contribution each one can make to the identity search challenge.

Edge.—The data set provides not only ~350k records, but also a pre-computed set of ~58k similarity-weighted edges based on the record attributes. These component (per attribute) and aggregated measures form a compressed static similarity assessment as an alternative to our other eight remaining complex similarity measures below. The "Edge" complex attribute is the total composite score of two records as recorded in this table. If the table does not specify an edge between two records, then their “Edge” similarity is zero.

Source.—Though we do not know the meaning of the various databases, combining information from different databases will be critical to transitive closure. Differences in record structure may reduce the similarity score based on substantive fields. To encourage exploration of cross-databases similarities, we assign a similarity component of 1 between two records if their sources are different, and 0 if they are the same.

Full Name.—A person’s name is the most specific and semantically meaningful identifier we have. Between two “Full Name” attributes we compute the similarity accumulative on the presence and match of the three component strings (0.5 for last name match, an additional 0.3 for first name match, and a final 0.2 for a middle name match). While our current implementation determines a name component match solely on (ignore-case) string identity, future extensions will compute the match using the Soundex algorithm [14] on each component of the name, to add robustness against spelling errors.

US State.—The “State” attribute of a record offers a rough approximation of the geographic location of this record. We define a static similarity measure between all states based on the normalized geographic distance of the latitude/longitude coordinates of their state capitals. Records with identical state identifiers of course have a similarity of 1. Records for states with the largest geographic distance of capital cities, or records that do not identify a (known) state, all have a similarity value of 0. Other similarity values are proportional in-between.

Zip.—Zip code (the US postal code system) offers a finer approximation of the geographic location of this record than the US-State attribute. Similarity between two Zip attribute values is first established by identity (similarity=1). If the Zip codes are not identical but both populated, their similarity is based on the [0, 1] normalized geographic distance of the latitude/longitude coordinates of their respective geodesic centers. Unpopulated records result in a similarity of zero.

Full Address.—The “Full Address” complex attribute combines our various geographic estimates of the location a record references. Complete or partial matches of the components of the Full Address accumulate similarity contributions. Matches for State and Zip entries are provided by their respective complex attribute wrappers. City and street name strings are either identical or not. If street names are identical, then highly similar house numbers provide a small similarity bonus.

Phone.—The only concrete identifier we have for the POI is a phone number, but the database contains some duplicate numbers, suggesting either shared phones or erroneous data. We accumulate units of similarity by first assessing the similarity of the area codes, then the prefix similarity, and finally the line code similarity. In this sequence, whenever there is not an identical match, further similarity accumulation will stop. For instance, there is no reason to compare line codes with different prefixes. If the area codes of two records already do not match, we use an external database of latitude/longitude coordinates of major cities or towns in the geographic coverage of the area code to provide a partial area-code similarity measure.

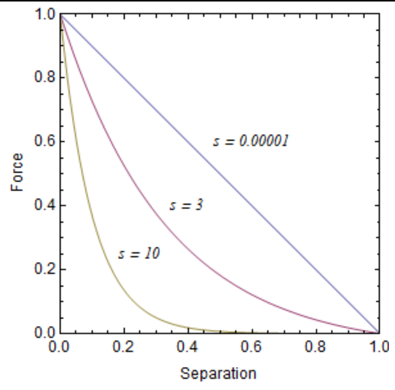

Figure 1: Effect of Shaping Factor *s* on Force

Gender.—The description of the challenge problem specifies that the POI is male. While there is no explicit “Gender” field in the databases, the first name of a person, if provided, does provide an estimate of the likely gender. We extracted a database of 48k international first names that provides an estimate of the likely gender. Gender attribution in this external database ranges takes one of 5 values: Strongly Female, Possibly Female, Unknown, Possibly Male, Strongly Male. Using string matching with the provided First Name attribute value, we assign each record one of these five gender identities. If no first name is provided, then Unknown is assumed. We found substantial coverage of the records in the challenge data set by this name database, once we added the gender identifiers of the fictional main characters of J.R.R. Tolkin’s Lord of the Rings trilogy. We manually defined a 5x5 similarity matrix between the gender identities, that assigns a similarity of 1 for matching Strongly Female or Male records and decreasing ranges of similarity for more distant fields in the gender identity ordering.

ID-DOC.—We do not know the meaning of the ID-DOC keys, but a cursory survey of the data shows that some of these are shared, both within and across databases. Binary string similarity (0 or 1) is assessed.

3.2.2 Force Definition

The force between two agents has two components: one repulsive and one attractive. Each force is computed using the distance scaling function

$$g(d,m,s) = \frac{e^{s\frac{m-d}{m}} - 1}{e^{s} - 1}$$

where

- d is the shortest distance on the torus between the two agents,
- m is the maximum distance on the torus,
- s is a shaping factor.

Figure 1 illustrates the effect of the shaping factor s. When s is close to 0, force decreases linearly with distance. As s increases, force drops off more rapidly.

For similarity φ between two agents, the force is

$$f(d,\varphi) = w_a\, d\, g(d,m,s_a)\varphi - w_b\, m\, g(d,m,s_b)\,(1-\varphi)$$

Figure 2 shows an example of this force for $s_r = 5$, $s_a = 6$, $w_a = w_r = 1$, $\varphi = .97$, showing repulsion at low separations, reducing force at large separations, and attraction in between. The force is multiplied by a maximum step length to give the distance that the agent moves in the direction of the agent with which it is being compared. The larger the step length, the more agents move on each iteration, but the more danger there is of thrashing.

3.2.3 Distributed Execution

We apply FBC repeatedly to selected pairs of points. Convergence is smoothest if step length is modest, which in turn requires each pair of points to be evaluated repeatedly. Application of this algorithm to a large number N of points thus involves $O(N^2)$ operations, which can be prohibitive for very large datasets.

In the experiments reported here, we distribute the force computation over multiple processes, which all access a centralized DB. The processes execute asynchronously. Each time a process is invoked, it

- Retrieves a cohort of c points and their current locations from the database (in our experiments, c = 35), of which most are randomly chosen, but a few ("queries") may be specified;
- Applies the FBC algorithm to all pairs of points in the sample, and computes new locations for them;
- Writes the points back into the database.

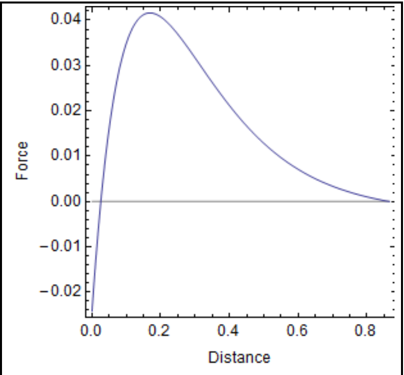

Figure 2: Example of computed force function

In addition, a process remembers the k closest agents to a query agent that it has seen, so the results of the clustering search can be retrieved by merging these lists across all processes.

FBC scales linearly in both space and time in the number of processors, and offers a quadratic benefit over a naïve algorithm, regardless of the degree of distribution. Let

- N = the number of agents;
- m = the number of processors;
- c = the number of points clustered by each processor at one time;
- d = the maximum step length an agent can take in a processing cycle, expressed as a fraction of the maximum distance D on the torus.

Consider the quadratic benefit first. A naïve approach computes the similarity of each agent to all other agents, for complexity $O(N^2)$. In FBC, an agent is closely attracted to a group of g other agents, and each interaction between two agents approximates g^2 interactions, reducing the complexity by a factor of $(N/g)^2$.

Processing of massive data is limited by both space and time, inviting distribution over multiple processors.

We expect time complexity to scale linearly with the number of processors. On average, assume that each agent starts out $D/2$ from its optimal location. Then an agent needs $O(1/(2d))$ interactions to reach its destination. For simplicity of analysis, assume that processes run synchronously. The probability of an agent being selected in a processing cohort is mc/N. Each selection yields c interactions, so an agent can anticipate mc^2/N interactions per processing round, thus requiring $N/(2dmc^2)$ processing rounds to move to its optimal location. Each round takes $O(c^2)$ time, so the total processing time is $N/(2dm)$. Interestingly, c does not affect time complexity, which scales linearly with the number of processors.

FBC processes run asynchronously, so two processes may simultaneously modify the location of the same point, and only the last one to write to the database will be preserved. The chance of a record being in a given clustering process is c/N. The chance that we get some record—any record—in two processes concurrently is $N*(c/N)^2 = c^2/N$. The number of possible pairs of processes is ${}_mC_2$, so the probability of collision is ${}_mC_2\, c^2/N$, which for c = 35, N = 350,000, m = 4 is about 4%. While c does not affect time complexity directly, it does affect collision probability, quadratically, commending a choice of small c. We do not attempt to detect these collisions, but rely on the incremental any-time nature of the algorithm to correct them over time.

Space complexity is also linear, since our clustering process needs hold in memory only the set of records being clustered. Empirically, we find that 350,000 records is at the limit of a single processor, while smaller cohorts are easily processed.

3.2.4 *Convergence Detection*

If steps are small enough, incremental distributed processes like FBC converge in a roughly exponential fashion to an asymptote [15]. To monitor convergence, we compare the pairwise separation of agents in the virtual space of the torus with their pairwise similarity. If we were able to capture all the similarity information in the spatial distribution, the rank ordering of distances between agents in space would be the same as their rank ordering in similarity. A natural measure of the correlation of these two orderings is the Kendall τ, defined as

$$\tau = \frac{P - Q}{\sqrt{(P + Q + T) * (P + Q + U)}}$$

The components of this metric assume a set of joint observations $\{(x_i, y_i)\}$ from two random variables X and Y. Consider two such pairs (x_i, y_i), (x_j, y_j). They can be related in four different ways, the first two of which are mutually exclusive with one another and with the latter two.

- P is the number of such pairs in which the rank ordering of $x_{\cdot}$ and $y_{\cdot}$ is the same.
- Q is the number of pairs in which the rank ordering is wrong: if $x_i < x_j$, $y_i > y_j$, or vice versa.
- T is the number of pairs in which the x values are tied.
- U is the number of pairs in which the y values are tied.

By construction, τ ranges from -1 if the variables are anti-correlated to +1 if they are perfectly correlated. We monitor the convergence of FBC by computing the pairwise separation of each pair of points with two metrics: similarity, and physical distance on the torus. The comparison of these two metrics is widely used to assess the effectiveness of multi-dimensional scaling, in the form of the Kruskal stress [10]. We then compute τ over this set of pairs of points as our measure of convergence. For efficiency, we consider a subset of pairs of points consisting of all pairs one of whose members is a query, and thus estimate the convergence of the distribution of points with respect to the query. Since retrieval in KBC consists of selecting those points that are nearest to the query, this measure accurately reflects the aspects of convergence that are important to us, ignoring dynamics far from the query that have no impact on retrieval.

4. COMPARISON WITH OTHER TECHNOLOGIES

The FBC algorithm competes with a number of other technologies, but also shares some of their features. We briefly review three: semantic analysis, cluster analysis, and multi-dimensional scaling.

4.1.1 Semantic Analysis

A semantic approach to identity matching would reason explicitly about the semantics of each available field, and seek to relate them symbolically. For example, it might represent the insight that if two people have the same address, they probably know one another. Such an approach is standard in classical artificial intelligence, and can bring a great deal of domain knowledge to bear on the problem. However, it is computationally very costly, and thus inappropriate for extremely large datasets. It also requires extensive knowledge engineering, slowing its application to problems that must be solved quickly.

FBC does take advantage of domain knowledge, in defining similarity metrics for complex attributes. The definitions discussed in Section 3.2.1 all incorporate our intuitions and semantic understanding of the problem. However, FBC translates these intuitions into numerical terms, permitting much faster computation than symbolic manipulation allows.

4.1.2 Cluster Analysis

Cluster analysis [3,6,9] seeks to identify entities that are near to one another by some measure. Centralized methods begin with a distance matrix giving the pairwise separation of entities, and many of them require updating this matrix repeatedly. Their complexity is thus at least $O(n^2)$, and in practice centralized implementations reach their limit with datasets on the order of 10^5 elements (e.g., 202,000 galaxies in [8]). Decentralized approaches [16] typically partition the data, cluster each subset separately, then exchange either cluster parameters (such as centroids) or representative points to estimate a merged clustering.

Cluster analysis, like FBC, seeks to discover nearby entities. However, it differs in three important ways.

- The need to construct and maintain a distance matrix means that it is difficult to apply to dynamic data. Typically, one cannot add data while clustering is going on. FBC is an any-time algorithm that can accept new data at any time (though our application in this project does not draw on this feature).
- Cluster analysis views entities as fixed in attribute space, and applies distance measurement to them as passive objects. FBC allows them to move in an abstract low-dimensional space, actively participating in their own organization.
- A consequence of the centralized distance matrix is that all attributes participate equally in global clustering decisions, hindering the analysis of heterogeneous data. FBC allows entities to interact pairwise, drawing only on those attributes that both entities possess. Integration across heterogeneous attribute sets happens by transitive interactions, in which an entity shares some attributes with one entity, other with another, and thus intermediates their interaction.

4.1.3 Dimensionality-Reduction Algorithms

The movement of FBC entities is reminiscent of some iterative-update algorithms for multi-dimensional scaling (MDS), an example of an important class of methods to reduce the dimensionality of a set of records. In general, dimensionality-reduction algorithms do not handle data that is massive, high-dimensional, dynamic, and heterogeneous. Algorithms for dimensionality reduction of distributed sensor data [5,18] rely on the homogeneous or near-homogeneous feature spaces of sensory data. Conventional schemes for dimension reduction, such as the linear FastMap algorithm [4] or nonlinear algorithms such as IsoMap [19] or Locally Linear Embedding [17], do not handle diverse or distributed data. Some of these schemes have been adapted to a distributed environment [1,11-13], but presume homogeneous data and a non-dynamic environment that allows iteration over static data collections on each processor. The common idea is to estimate locally which data items are relevant globally, and exchange these estimates iteratively across the network of processors. In high-dimensional data, not all dimensions are relevant to every interaction. Structure among subsets of the data lies on a much lower-dimensional manifold, whose dimensions typically depend on the query. Systems that exploit this insight [2,7] require access to all the data in a single process, and so do not support the distribution needed for timely processing of massive data.

5. EVALUATION

In the following, we discuss preliminary results from experiments with the data set, using only the “Edge” similarity coefficient, and provide an initial assessment of the convergence characteristics of the information matching process with a small artificial data set.

5.1 Results

The information matching process is inherently parallel and can be distributed for essentially linear (with the number of processors) performance gains over large scale networks and potentially deployed into a MapReduce/Hadoop cloud-computing environment. The experiment reported here used three standard WinTel PCs to execute 4 clustering processes each and an additional PC to run the MySQL database with the 350k records and their clustering coordinates. In this small setup, we arrived at the results reported here in less than two days execution even though one PC (4 processes) failed due to network problems after less than 8 hours.

The clustering space is a unit (1x1x1) box with all its 6 faces wrapped. Thus, we can operate in a finite volume without having to address edge-effects. The challenge data set is small enough to be visualized in its entirety in a centralized user interface. Figure 3 shows the raw result. It highlights the common location of the three query records (phone, address, phone+address) in the upper right corner. Adjacent to the queries, the information matching process highlights a relatively small set of nearby neighbors (data records) as relevant. This small set is clearly separated by a “Moat” from the rest of the data. Because the faces of the Clustering Space are wrapped, there is a small sphere of relevant data records near the queries, then an empty volume, and then a larger sphere of irrelevant records filling the remaining volume.

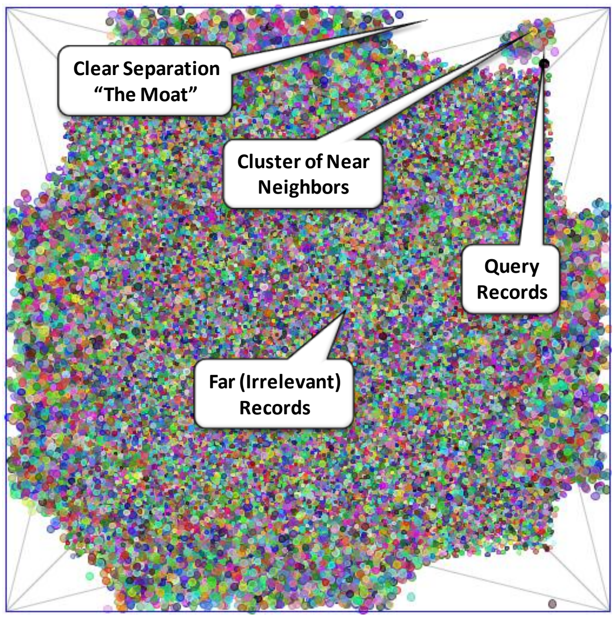

Figure 3: Converged Location of all 350k Records in Clustering Space

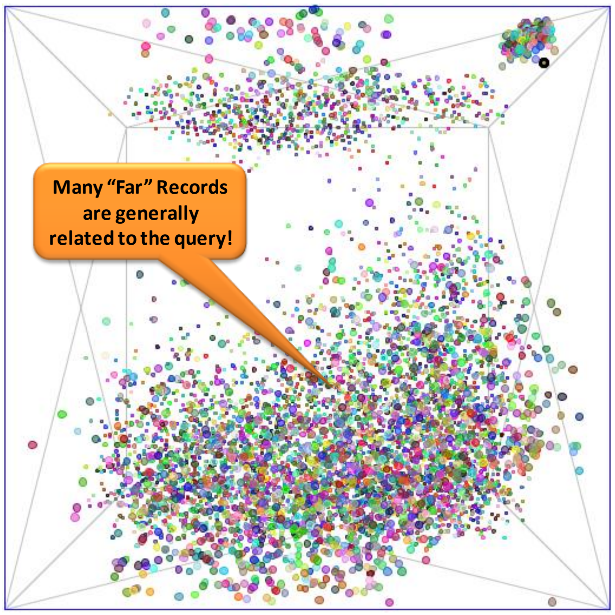

Figure 4: Converged Location of 7k Generally Relevant and Connected Records in Clustering Space

Does the outer volume of "irrelevant" records just contain unstructured noise? This question is answered by Figure 4. There, we only draw the location of records that are either queries or that meet the following constraints:

- They must have edges in the challenge data edge table
- They must either set "NY" as their State, "Bethesda" as their city, or "212" as their area code.

7,195 records in the challenge data meet these constraints and, given the nature of the challenge, one would consider them generally relevant to the identity discovery problem. Inspection of the cluster arrangement shows that many of these records are far from the query. Thus, the information matching process is highly selective. Furthermore, we find that there is clearly structure in the outer volume (rather than just uniform distribution), which is encouraging when one wants to be able to process more than one query at a time.

Does the experiment substantiate our "transitivity" claim, or does the close neighborhood of the query records only contain explicitly similar records? In Figure 5, we plot for all pairs of data record and query record from Figure 4 the explicit edge similarity of the pair over their distance in clustering space. We recognize the gross structure of the arrangement, with a large number of records far away from the queries (all without direct similarity), the empty space in the mid-region, and the cluster of relevant records near the queries. In the fine structure of the query neighbors, we find that data records with non-zero direct similarity are ordered in clustering distance according to their similarity with less similar records farther away than more similar ones. But, in confirmation of our transitivity claim, we also find a significant number of data records near the query records that have no direct similarity to those queries. Those have been pulled in by the transitive nature of our force-based clustering.

Finally, to the most important question in the experiment: Do we have an answer that can be explained? To answer this question, we constructed a rudimentary user interface, connected in real-time to the database, that supports the exploration of the information-matching results. We select one of the query records, which triggers the collection of a few of the nearest neighbors to the query agents in clustering space. Those are shown in the second list, ordered by their cluster-space distance and with a retrieval match estimate derived from their normalized (by maximum possible distance in space) distance to the selected query record. Selecting any neighbor from that list triggers an excursion into the underlying attribute-similarity space. In an exhaustive recursive process starting at the query record, we are

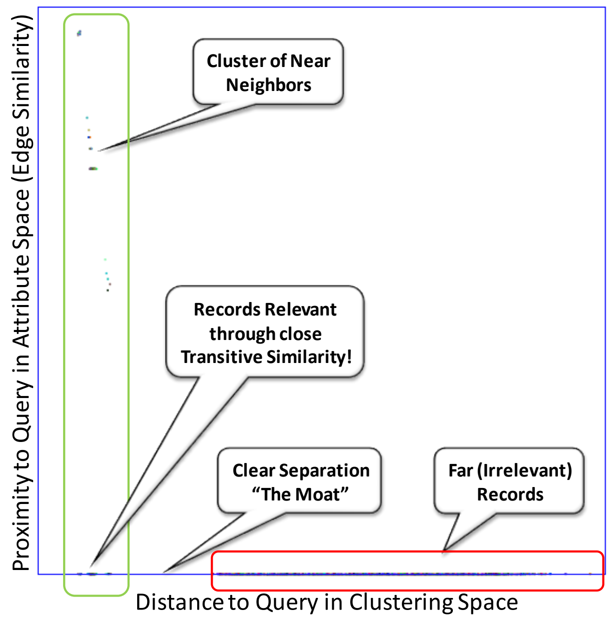

Figure 5: Correlation Between Clustering Distance (x) and Pair-Wise Edge Similarity of Data Records to Query Records

looking for non-trivial (more than one step) transitive paths that lead to the selected data record. Due to the high node degree of the underlying similarity graph, the combinatorial explosion of this search limits us to consider only paths up to 5 steps. But fortunately, a data record with more than 92% similarity to the query record based on distance in clustering space and ranked #6 among all neighbors of this query, is connected to the query through only two intermediate records (note that these records themselves do not have to be close to the query in clustering space). Thus, we present our answer to the challenge problem in the form of this (embellished) explanation, following the path of records from the (incomplete phone number) query to the 6th-nearest neighbor:

- The unidentified male "Mr. X" lives in New York with his wife, Gloriana D. Brandybuck. They share a land-line phone number consistent with the clinic records at their home at 3306 Rosewood Lane, New York, NY 10003.
- Since they have just married, Gloriana Donnamira Brandybuck's driver's license or credit card (a Doc-ID) is still registered with her old address at 2719 Pin Oak Drive, Manhattan, NY 10018.
- They traveled to Maryland and checked them into a hotel at 18 Wayback Road, Bethesda, MD 20014, using the same Doc-ID.
- The day after their arrival, Mr. X fell ill and decided to visit the medical clinic nearby at 4408 East Madison Ave., Bethesda, MD, 20014.

The transitive chain starts with a match on an incomplete phone number, which provides a name (and first address), which links us to another record with a similar name (and a second address) and provides us with a Doc-ID, which leads to an address near the medical clinic. After we presented this conclusion, the sponsor who provided the data verified that Mr. X is indeed the husband of Gloriana Brandybuck.

5.2 Convergence Assessment

We claim that FBC can be distributed over many processing units by aggressively sub-sampling the number of clustering interactions that have to be computed. We also claim that this distributed process has exponential convergence characteristics that provide a good answer fast and improved answer if more time is available (any-time characteristic). To assess these claims, we performed an initial experiment with an artificial data set of 350 color (RGB) data records, groups into seven clusters, and one query record. We start the experiment with a random arrangement of the records' agents in cluster space and run to (manually determined) convergence (Figure 6).

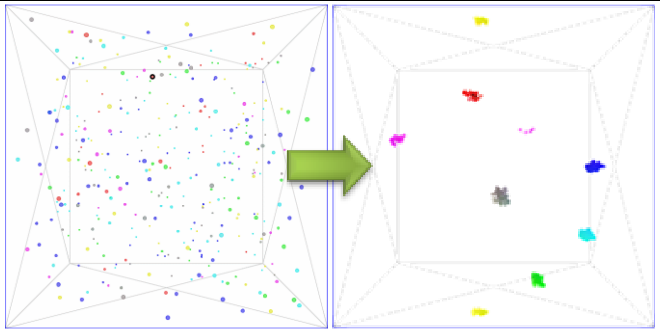

Figure 6: Information Matching with 351 Color Records.

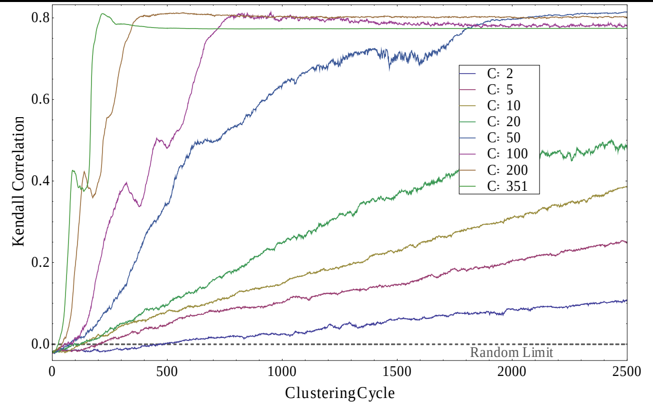

Figure 7: Kendall Correlation over Clustering Cycle.

We repeat the experiment for varying values of *c*, which determines in these single-processor runs, how many randomly selected agents are allowed to interact in each cycle. Thus, we emulate a distributed execution, where *c*/351 corresponds to the number of parallel processes we deploy. Our sequential execution of the random sampling ignores the possibility for collisions (the same record moved by more than one process at the same time), which will introduce additional noise.

Figure 7 plots the evolution of the Kendall Correlation measure (Section 3.2.4) – the quality of our clustering of data relative to query record(s) – over actual execution cycles for all 8 tested values of *c*. As expected, the clustering requires more cycles (moves of sub-sets of size c) with smaller *c*. But we already recognize in the shape of the curves for larger *c* (>10), that the converged state is indeed approximated exponentially.

Figure 8 assesses the impact of parallelization by scaling the x-axis for each data series by a factor of *c* and correcting for the movement of the query record. Thus scaled, the convergence curves trend very closely to each other, suggesting that a nearly linear speed-up with the number of processors may be accomplished in a distributed setting.The additional noise introduced by the sub-sampling with small *c* actually improves the final quality of the converged result. We hypothesize that, similar to simulated annealing processes for instance, the noise prevents the clustering from falling into sub-optimal stable states and

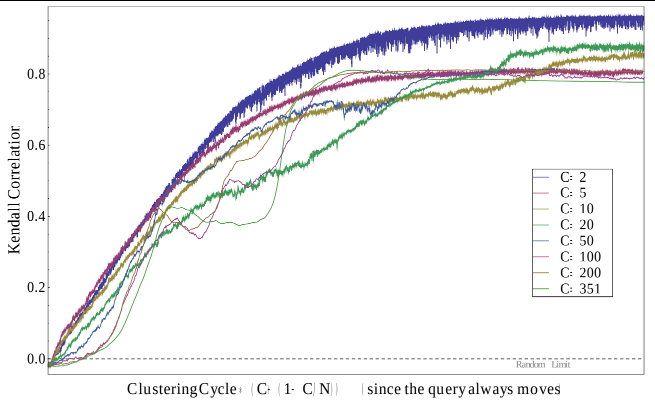

Figure 8: Kendall Correlation over Clustering Cycle * *c*.

instead drives it closer to the global optimum.

6. POTENTIAL EXTENSIONS

The current project demonstrates the ability of FBC to find transitively related groups of records in a distributed environment that can scale to handle massive data. There are numerous opportunities for extension.

Weight assignment.—The weights of similarity components critically affect the clustering behavior. We have assigned them based on our intuitions about the relative importance of various complex attributes, and in fact our initial experiment uses only one attribute, based on the edge file. It would be useful to estimate them in a more disciplined fashion, based on continuous analysis of actual databases against possible identity matching challenges.

Human interaction.—The system currently simply reports the closest *k* records to a query for further human processing. Ideally, humans would interact with the system iteratively as it runs. Several modes of interaction are possible:

- When two records persistently stay very close together, we should entertain the hypothesis that they represent the same person, and merge them, thus reducing N and speeding up processing. A human monitor could review and decide on such recommendations.
- The dynamics of the system may indicate that certain records are moving erratically, depending on the other records with which they interact. Such differences may arise from different sets of attributes that the various groups of records hold in common, and might motivate prioritized information requests to collect more data on a specific record.
- A human overseer might change the weights of different similarity components (e.g., increase effect of address, decrease effect of name), thus exploring different dependencies among the components of the problem.

Data distribution.—The system currently distributes processing, but uses a single database. It can be extended to distributed data using HADOOP, providing even greater scalability.

Dynamic data.—The system currently works with a fixed set of records. However, the FBC algorithm readily supports the introduction of new records as it is running, simply by dropping them into a random location on the torus. This capability opens the potential to running a continually self-organizing database into which new records can quickly be introduced.

7. CONCLUSION

Many important applications in epidemiology and domestic security require the ability to discover transitive linkages across heterogeneous databases rapidly, without reasoning explicitly about possible scenarios. Instead of reasoning about the various records, Force-Based Clustering (FBC) turns each record into a software agent that moves in an abstract information space in response to the net “force” it feels from other agents. These forces in turn are defined by generic similarity measures over commonly occurring fields, measures that can readily be defined in advance and applied quickly to available information. The agent interactions can be distributed over many processors to speed the clustering process. Application of this approach to a synthetic data set (provided by an anonymous sponsor external to our research group) allows us to identify the person of interest.

8. REFERENCES

[1]F.N. Abu-Khzam, N. Samatovaz, et al. Distributed Dimension Reduction Algorithms for Widely Dispersed Data. In *Proc. the Fourteenth IASTED International Conference on Parallel and Distributed Computing and Systems (IASTED PDCS 2002)*, pages 167-174, ACTA Press, 2002.

[2]C.C. Aggarwal, P.S. Yu. Finding Generalized Projected Clusters In High Dimensional Spaces. In *Proc. SIGMOD Conference*, pages 70-81, 2000.

[3]P. Arabie, L.J. Hubert, et al. *Clustering and Classification*. Singapore, World Scientific, 1996.

[4]C. Faloutsos, K.-I.D. Lin. FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets. In *Proc. ACM SIGMOD*, pages 163-174, 1995.

[5]J. Fang, H. Li. Optimal/Near-Optimal Dimensionality Reduction for Distributed Estimation in Homogeneous and Certain Inhomogeneous Scenarios. *IEEE Transactions on Signal Processing*, 58(8):4339-4353, 2010.

[6]J. Hartigan. *Clustering Algorithms*. New York, NY, John Wiley and Sons, 1975.

[7]A. Hinneburg, C. Aggarwal, et al. What is the nearest neighbor in high dimensional spaces? In *Proc. 26th Int. Conf. on Very Large Data Bases (VLDB 2000)*, pages 506-515, Morgan Kaufmann, 2000.

[8]W. Jang, M. Hendry. Cluster analysis of massive datasets in astronomy. *Statistics and Computing*, 17(3):253-262, 2007.

[9]L. Kaufman, P.J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. New York, John Wiley & Sons, 1990.

[10]J.B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29:1-27, 1964.

[11]P. Magdalinos, C. Doulkeridis, et al. K-Landmarks: Distributed Dimensionality Reduction for Clustering Quality Maintenance. In *Proc. 10th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'06)*, 2006.

[12]P. Magdalinos, C. Doulkeridis, et al. A Novel Effective Distributed Dimensionality Reduction Algorithm In *Proc. SIAM Feature Selection for Data Mining Workshop (SIAM-FSDM'06)*, 2006.

[13]P. Magdalinos, M. Vazirgiannis, et al. Distributed Knowledge Discovery with Non Linear Dimensionality Reduction. In *Proc. the 14th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'10)*, 2010.

[14]NARA. The Soundex Indexing System. National Archives and Records Administration, 2007. http://www.archives.gov/research/census/soundex.html.

[15]H.V.D. Parunak, S.A. Brueckner, et al. Global Convergence of Local Agent Behaviors. In *Proc. Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS05)*, pages 305-312, ACM, 2005.

[16]A. Rajaraman, J.D. Ullman. *Mining of Massive Datasets*. Cambridge, UK, Cambridge University Press, 2011.

[17]S. Roweis, L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science* 290(5500):2323--2326, 2000.

[18]O. Roy, M. Vetterli. Dimensionality Reduction for Distributed Estimation in the Infinite Dimensional Regime. *IEEE Trans. Information Theory*, 54(4):1655-1669, 2008

[19]J.B. Tenenbaum, V.d. Silva, et al. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290(December 22):2319-2323, 2000.